Universitat de Lleida

Escola Politècnica Superior

Grau Enginyeria Informàtica

Treball Final de Grau

Generació de contingut web accessible per a tothom: cas d'un Plug-in per a WordPress

Eric Labara Olvera

Director/a: Toni Granollers i Jonathan Chiné

Juliol, 2016

Índex

Índex d'il·lustracions

1. Introducció

1.1 Resum

El treball que s'ha realitzat durant els últims mesos parteix d'una aplicació web sorgida a partir de la tesi doctoral de la doctora Afra Pascual titulada "Accesibilidad en entornos web interactivos: superación de las barreras digitales" i que pot trobar-se sencera en aquesta adreça (http://www.tdx.cat/handle/10803/314581) del portal web Tesis Doctorals en Xarxa. Aquesta tesi explora la problemàtica en la publicació de continguts web no accessibles, ja sigui involuntàriament o per falta de coneixement del publicador. En finalitzar tot el procés es genera una aplicació web anomenada "Emphatic Editor for Accessibility" (EE4A) la qual mitjançant el diàleg amb l'usuari, l'aconsella i ajuda a reparar els errors d'accessibilitat.

En aquest punt entra en joc aquest treball final de grau, l'objectiu principal és la creació d'un Plug-in per a un popular gestor de continguts, el qual ens permeti realitzar aquest diàleg creat en la web EE4A en el moment de publicar cada entrada al sistema gestor de continguts, amb la finalitat que el contingut publicat sigui ja accessible. Per a poder fer tot això possible, durant els mesos de desenvolupament s'ha modificat i adaptat la web original per a poder crear una API en el servidor. Posteriorment s'ha desenvolupat una aplicació client en AngularJS, que funciona com a Plug-in de WordPress, i que consumeix aquesta API.

L'objectiu d'aquesta aplicació és ser executada en l'entorn WordPress per a poder aplicar tota la seva funcionalitat d'una manera útil dins aquest entorn, permetent-nos així reparar els errors d'accessibilitat que poden sorgir en les pàgines creades pels creadors de contingut WordPress.

1.2 Objectius del treball

En aquest apartat es descriuen la llista d'objectius que m'he marcat per tal d'assolir l'objectiu principal del projecte. Aquest objectiu és crear un generador de contingut web accessible, que consisteix en aconseguir la creació d'un Plug-in per al gestor de continguts WordPress que ens permeti comunicar-nos amb l'aplicació web EE4A i poder realitzar tots els processos d'avaluació i reparació necessaris per a poder generar el contingut accessible directament a l'hora de publicar.

Entendre i continuar el desenvolupament d'una aplicació rebuda

En el cas d'haver de continuar un treball amb una base ja existent, és a dir, que ja hi ha un llarg treball anterior al punt d'inici, l'adaptació als requeriments imposats i el procés de comprensió del material obtingut és una part realment important. Per tant aquest objectiu engloba la capacitat d'adaptació i comprensió envers la situació de rebre una web completament funcional creada per una altra persona i feta amb un llenguatge i framework desconeguts. A més, dins de l'objectiu s'inclou el procés d'entendre el treball realitzat anteriorment en el projecte, el recorregut que els ha portat fins a aquest punt i la finalitat que es pretén aconseguir.

La tesi doctoral d'on va sortir el projecte s'estudiava la problemàtica en què milions de persones sense coneixements tècnics publiquen contingut en la Web, blocs, wikis i xarxes socials, tot i que existeixen recomanacions d'accessibilitat de W3C, els usuaris, inconscientment, segueixen publicant continguts que presenten barreres a les persones amb discapacitat. La tesi explora aquesta problemàtica i, amb la intenció de solucionar-la, posa el focus en la comunicació de les barreres d'accessibilitat a les persones que publiquen contingut a la Web sense coneixements tècnics. La hipòtesi que fonamenta la tesi és que «reduint la complexitat de la informació relacionada amb l'accessibilitat, es propiciaria l'aplicació de criteris d'autoria accessibles, augmentant la qualitat general del contingut web».

Crear un Plug-in de WordPress

S'ha de crear un Plug-in seguint els estàndards i directives del gestor de continguts WordPress, el qual ens permeti carregar codi JavaScript per a realitzar les operacions necessàries i la comunicació amb l'aplicació web

Crear API Rest

Crear un mòdul en l'aplicació "Emphatic Editor for Accessibility" el qual sigui una API Rest per a poder comunicar-nos via HTTP entre el Plug-in i l'aplicació. Fer l'API seguint els estàndards habituals per facilitar l'operativitat per a futures ampliacions.

Tests

Un cop finalitzat el procés de producció s'hauran de realitzar diversos tests per comprovar que realment l'aplicació està fent el que es demana. A més, s'hauran de realitzar diversos tests d'usuari per comprovar que el procés és fàcil i entenedor.

1.3 Introducció i motivació

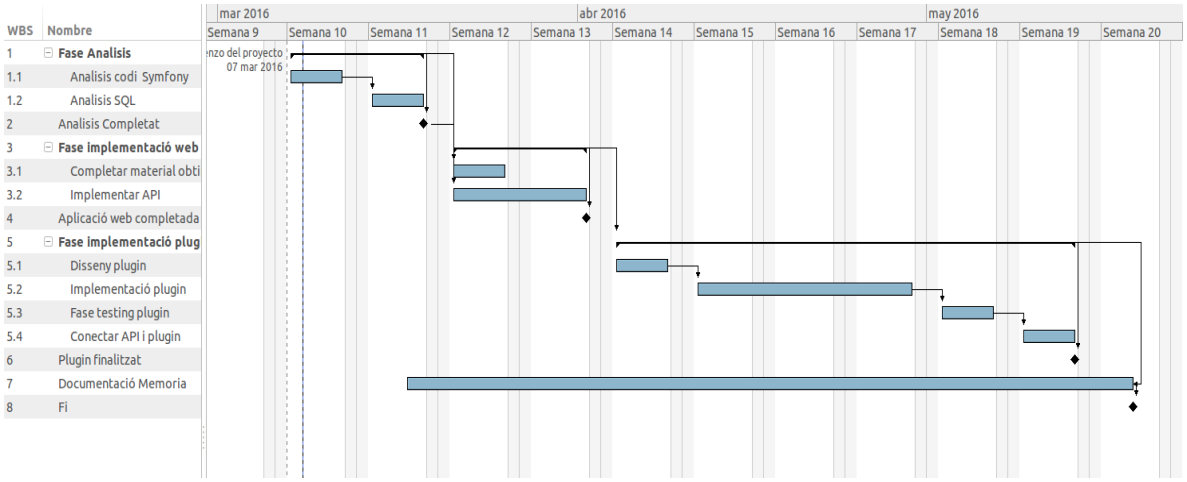
Aquest treball realitzat sorgeix d'una proposta del Grup de Recerca en Interacció Persona-Ordinador i Integració de Dades, GRIHO, els qual esta basat en prosseguir el treball originat a partir de la tesi doctoral de la doctora Afra Pascual Almenara (Pascual Almenara, 2015). Personalment em vaig posar en contacte amb el director de la proposta, Toni Granollers.

Personalment, buscava un treball en el qual pogués desenvolupar una aplicació, ja que el meu mòdul està enfocat a l'enginyeria del software, per tant aquest treball encaixava. Seguidament volia un treball el qual m'ajudes a aprendre coses noves, en aquest cas PHP o el framework Symfony, per poder-me endur alguna cosa més, alguna cosa nova a part de l'experiència adquirida. A més, el repte de realitzar una aplicació web partint d'una base, era un repte que m'atreia, ja que segurament al llarg de la meva vida professional hauré de lidiar amb aplicacions ja existents fetes per una altra persona. Aquests punts que he esmentat són principalment els factors purament tecnològics més ben dits pràctics que m'han portat a elegir aquest tema com a treball final de grau.

Per un altra banda, personalment, m'agrada realitzar aplicacions les quals, més que la funcionalitat en si, puguin aportar alguna cosa diferent. En aquest cas l'aplicació que he realitzat, si es continua treballant i perfeccionant, pot arribar a ser una aplicació molt útil perquè totes les persones que publiquen contingut web en l'entorn WordPress, puguin fer-ho publicant contingut accessible per a persones amb diferents discapacitats, sense la necessitat de tenir cap tipus de coneixement sobre accessibilitat web. En cas de què el projecte segueixi i l'aplicació es publiqui, els beneficiats en seran les persones que tenen aquestes discapacitats, ja que ara podran accedir a aquest tipus de continguts web que anteriorment els podia suposar una barrera.

1.4 Temporalitat

En la fase inicial del projecte es va definir un calendari amb les tasques a realitzar durant el transcurs d'aquest projecte. En el moment inicial el calendari era el següent:

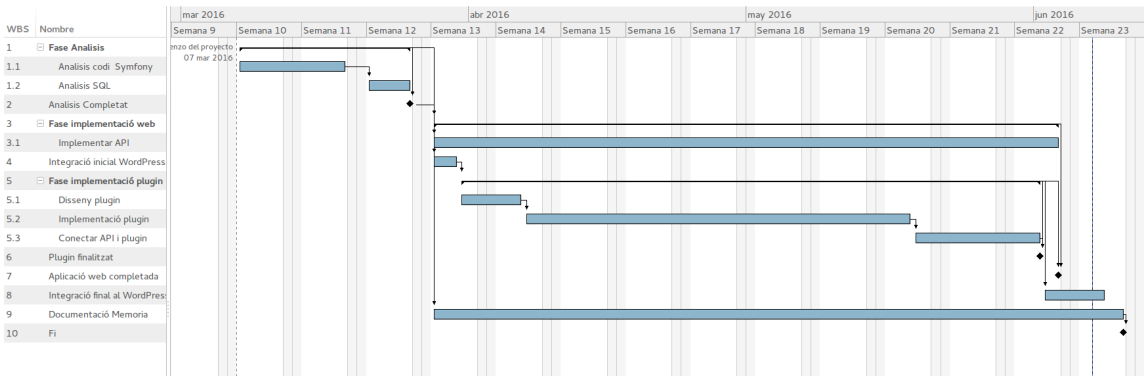


Il·lustració 1. Planificació temporal inicial

En aquesta planificació inicial s'estimaven 3 fases de desenvolupament, anomenades: anàlisis, implementació web i implementació Plug-in. El projecte es va iniciar el dia 7 de març i es preveia finalitzar-lo aproximadament l'ultima setmana de Maig de 2016.

Posteriorment, quan el desenvolupament ja estava iniciat van anar sorgint problemes inesperats els quals van obligar a canviar l'ordre de les tasques a realitzar. L'explicació d'aquests contratemps, i com van ser gestionats seran descrits més endavant en la secció Gestió del Projecte.

Així doncs després de completar el desenvolupament la planificació final ha set la següent:



Il·lustració 2. Planificació temporal final

Entre la planificació inicial i la planificació final es poden veure grans canvis (Es poden veure les imatges amb major dimensió a l'Annex 1).

El primer gran canvi que s'aprecia és que la implementació de l'API es realitza simultàniament amb el desenvolupament del Plug-in. Quan en un punt del projecte sorgien problemes en el client i no es podia avançar es canviava a desenvolupar servidor i en canvi si hi havia problemes en el servidor es canviava al client.

Els altres aspectes que es poden apreciar són la inclusió de la tasca integració en el WordPress la qual no s'havia tingut en compte anteriorment. A part d'això també trobem que la durada de moltes tasques ha canviat substancialment. En la primera estimació s'estimava acabar la sobre la setmana 21 i en canvi s'ha acabat sobre la setmana 24. Aquest augment de temps ha set perquè les tasques s'han demorat en el temps sigui per falta de previsió o problemes sorgits en el projecte.

Alguns exemples d'aquest canvi és la fase d'anàlisi, i sobretot la fase d'implementació, ja que han augmentat bastant el seu temps de realització. La inclusió de la fase d'integració també ha contribuït, això si, en menor manera a l'increment total de temps.

El cost d'implementació del servidor també ha augmentat, lògicament al ser desenvolupat concurrentment amb el client. Òbviament el cost temporal de desenvolupar el servidor no es igual, ja que s'ha anat desenvolupant per seccions, quan era necessari, per tant el pes de desenvolupar el servidor és molt inferior al mostrat.

1.5 Pressupost

Anàlisi

Concepte	Hores de treball	Preu/Hora	Total
Anàlisi Symfony	40	10€	400
Anàlisi SQL	20	10€	200
			600€

Disseny i Implementació

Concepte	Hores de treball	Preu/Hora	Total
Disseny Servidor	12	25€	300
Implementació servidor	80	15€	1.200
Disseny Client	20	25€	500
Implementació Client	160	18€	2.880
Integració en el entorn WordPress	24	15€	360
			5.240€

Altres

Concepte	Hores de treball	Preu/Hora	Total
Documentació	60	10€	600€
Costos materials (Llum, ordenadors)	-	-	400€
Llicències de programari usat per a desenvolupar	-	-	175€
			1.775€

TOTAL	7.015€

2. Bloc de Tecnologies Relacionades i estudis realitzats

2.1 Tecnologies usades en el projecte

PHP i Symfony2 Framework

L'aplicació web "Emphatic Editor for Accessibility" està programada amb el llenguatge de programació PHP i utilitzant el framework Symfony en la seva versió 2.

Symfony és un popular framework PHP enfocat a la creació d'aplicacions web implementant el patró MVC . Els principals punts forts són la seva rapidesa, el baix consum de memòria, i la seva alta flexibilitat. Dins del framework tot està organitzat en "Bundles" que serien paquets on es poden implementar les diferents funcionalitats de l'aplicació web.

Una alternativa a Symfony seria CodeIgniter que també és un framework MVC, però en aquest cas és molt senzill amb una documentació molt clara i configuració simple, la principal diferencia amb Symfony seria que CodeIgniter és molt més senzill i en canvi Symfony molt més complex, estructuralment parlant. Una altra alternativa seria utilitzar un framework com Laravel, ja que integra nativament la possibilitat de crear aplicacions RESTful, mentre que en Symfony depens de les extensions.

L'elecció d'aquesta tecnologia ha estat imposada pel fet de rebre aquesta aplicació web ja realitzada en aquest framework. En aquest cas el procés d'aprenentatge ha estat difícil i molt més lent de l'esperat a causa de les dificultats en trobar informació envers al framework. Tota la informació sobre el framework la vaig obtenir de la documentació oficial (Documentation: Symfony, 2016) i del llibre (Eguiluz, 2012). Tot i això, la documentació estava realitzada en àmbit general i em va costar una mica trobar informació més especifica quan em sorgia algun error.

La majoria de dificultats que em van sorgir en aquest punt, com he mencionat anteriorment, estan relacionades amb l'estructuració del framework el qual resulta certament complex. Per a poder tirar endavant he hagut de recórrer bàsicament al llibre mencionat anteriorment, (Eguiluz, 2012), ja que la documentació oficial al meu parèixer era molt simple i breu.

En referencia al llenguatge de programació, PHP, no hi va haver gaire dificultat, ja que és un llenguatge molt estès en la programació web. Qualsevol dubte que em sorgís estava perfectament explicat en la documentació oficial, (PHP Manual, 2016)

MySQL Database

MySQL és un popular sistema gestor de base de dades molt utilitzat en aplicacions web. La raó per la seva utilització i no la d'un altre sistema gestor de bases de dades com podria ser PostgreSQL és la seva rapidesa, la seva predefinida integració en PHP i el fet que forma part de les piles de desenvolupament XAMP i LAMP.

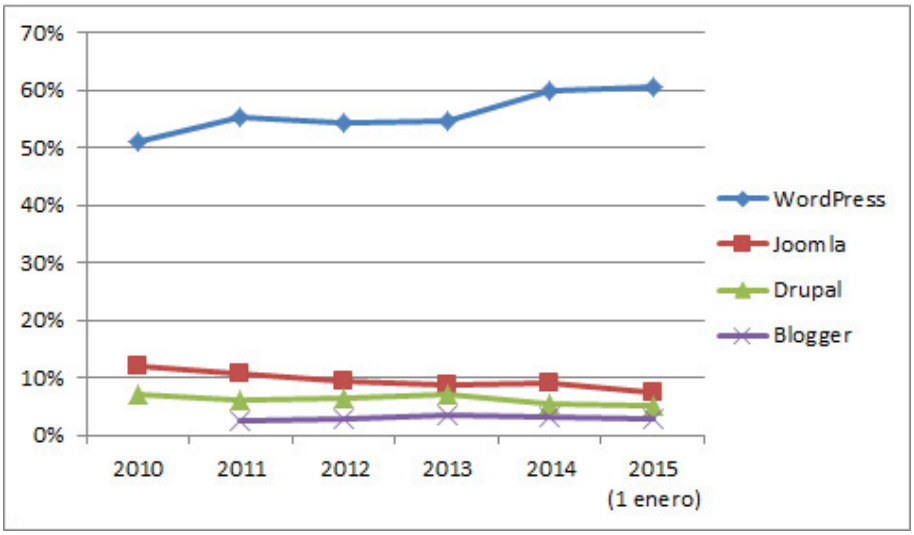
Es podria incloure dins de l'apartat anterior. Pràcticament no s'ha hagut de treballar ni modificar res de la base de dades de l'aplicació, però sí que eren necessaris els coneixements bàsics per a la instal·lació i petites modificacions per al correcte funcionament de l'aplicació.

En aquest cas no s'ha necessitat gaire documentació, ja que personalment ja coneixia aquest sistema de bases de dades i tampoc s'ha hagut de modificar res essencial. (MySQL, 2016)

WordPress

WordPress és un sistema gestor de continguts creat amb PHP i MySQL (WordPress, 2016) És open-source i és considerat el CMS més popular en la web actualment. Uns altres populars gestors de continguts serien Drupal i Joomla.

La principal raó per la utilització de WordPress és el fet que és el més popular i més utilitzat, per tant l'aplicació podrà arribar a un nombre més gran d'usuaris. A més a més en ser un CMS enfocat a Plug-ins ens dóna moltes facilitats per a la integració de noves funcionalitats.



Il·lustració 3. Percentatge d'ús de CMS

Per poder implementar una funcionalitat extra en el gestor de continguts WordPress s'han de seguir una sèrie d'estàndards i directives marcades pel mateix gestor, per a poder instal·lar i crear el teu Plug-in o per poder comunicar-te amb les funcions essencials del gestor de continguts. Per aplicar aquesta tecnologia correctament m'he hagut de basar amb la documentació de WordPress especialment enfocada a la creació d'extensions. La documentació ha resultat de molta utilitat, ja que es força completa tot i que a vegades no ho soluciona tot. (WordPress Developers, 2016)

AngularJS

El framework AngularJS és un framework complet MVC basat en JavaScript orientat a la programació d'aplicacions web. En l'actualitat AngularJS és dels més utilitzats i un dels que hi ha més informació en la xarxa a causa de la seva popularitat. Altres alternatives com Backbone.js són considerats més com a llibreries que com a framework complets com Angular. (AngularJS, 2016)

La part amb més pes i més important del projecte se l'emporta el client i la decisió de la utilització d'aquest framework ha estat personal, ja que tenia una mica d'experiència prèvia i coneixia la seva potencia per a poder realitzar les operacions necessàries. Es podria haver realitzat amb PHP o JavaScript directament, però perquè Angular? A simple vista pot semblar una feina breu i simple, però al contrari. Pràcticament s'ha d'implementar una web sencera amb les seves múltiples vistes, amb connectivitat via API, que pugui ser integrada i executada únicament com a un modal. A demès al mateix temps ha de funcionar basada amb una màquina d'estats (ignorant les URL per no molestar el gestor de continguts).

Tota aquesta funcionalitat es pot implantar directament amb JavaScript, però crearíem un Plug-in difícilment comprensible i difícil de mantenir en un futur. Per tant coneixent la seva potència i compensant el pes que em suposava ja l'aprenentatge en la part del servidor vaig decidir decantar-me per la utilització d'aquest framework.

Com he dit, ja coneixia força l'entorn i tota la informació que he necessitat l'he obtingut a través de la documentació oficial (AngularJS Documentation, 2016). En ser la part amb més treball en el projecte, aquesta documentació ha estat la més utilitzada amb diferència per resoldre qualsevol petit dubte que sorgís durant el desenvolupament.

2.2 Estudis realitzats: La Tesi doctoral

Com he dit anteriorment, Accessiblidad en entornos web interactivos: Superación de las barreres digitales: Així es titula la tesi doctoral de la doctora Afra María Pascual Almenara. Aquest va ser el punt d'inici del present projecte. En la tesi doctoral s'estudia el fet que persones sense coneixements tècnics estan publicant continguts en entorns webs sense saber que estan publicant continguts no accessibles. La hipòtesi de la tesis es basa en què si es reduís la complexitat de la informació relacionada amb l'accessibilitat, es reduiria la publicació de contingut no accessible. Amb aquest objectiu va sorgir l'aplicació EE4A la qual posa persones fictícies darrere de les barreres, comunicant les dificultats que tenen a l'hora d'accedir al contingut web que s'ha avaluat anteriorment en l'aplicació.

Una barrera, per definició, és un obstacle que impedeix o dificulta la realització d'una tasca. En entorns web interactius aquestes barreres impedeixen i dificulten que les persones amb algun tipus de discapacitat puguin accedir als continguts que ells desitgen.

En aquest camp el professor Giorgio Brajkinc de la universitat de Udine, va realitzar un projecte anomenat Barrier Walkthough Method, en el qual les barreres que possiblement poden afectar a les diferents discapacitats es relacionen amb les pautes WCAG.

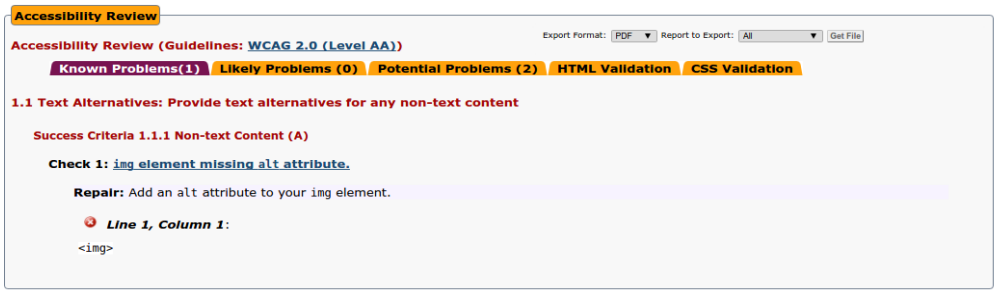
RICH IMAGES LACKING EQUIVALENT TEXT

Users:	Blind persons
Group:	Perception
Wcag 1.0:	1.1
Wcag 2.0:	1.1 1.1.1
Italian law:	3
Cause:	The page contains some image that provides information (e.g. a diagram, histogram, picture, drawing, graph) but only in a graphical format; no equivalent textual description appears in the page.
Failure mode:	The user, even if s/he perceives that there is an important image, has no way to get the information it contains. In addition s/he spends time and effort trying to find out where in the page or site that information is buried.
Effect:	The user cannot use the information conveyed by the image. Significant reduction of effectiveness; also user productivity can be reduced.
Fix:	Add an equivalent textual description to the image: by using the ALT attribute of IMG, and if not sufficient by using the OBJECT tag and specifying the text in the content of the tag. Should this still be insufficient, then add a link in the vicinity of the image that leads to a specific page where the textual description is present. Another strategy is to place the equivalent text close to the image so that it can be seen also by those who can see the image.

Il·lustració 4. Project Barrier Walkthough Method (Giorgio Brajknic)

Sabent tot això ara només ens fa falta poder obtenir la pauta o "check" WCAG d'una pàgina HTML per a poder relacionar-la amb una barrera d'accessibilitat, i per a realitzar aquesta tasca s'utilitza l'avaluador Achecker

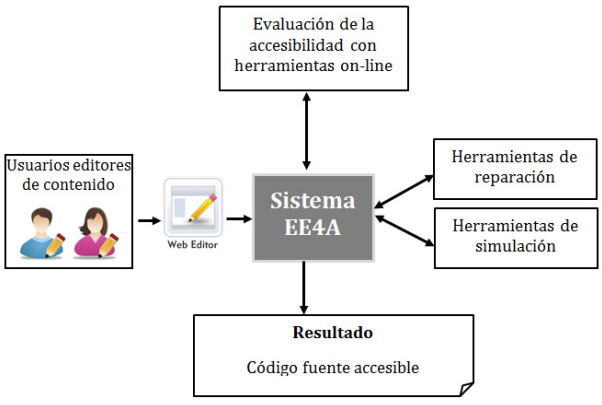
L'Achecker és una eina que comprova les pàgines HTML per avaluar si estan realitzades d'acord als estàndards d'accessibilitat i poden ser consumides per tothom. En el moment de realitzar una avaluació amb l'Achecker ens retorna una informació similar a la següent.



Il·lustració 5. Avaluació Achecker

Una vegada generada l'avaluació, a partir d'aquests "checks", ja podem determinar quines barreres estan afectades i com solucionar els errors d'accessibilitat. Aquesta és la base i un dels punts més importants del projecte.

Arribat en aquest punt, ja tenim les persones i les barreres afectades. Ara només s'ha de generar un diàleg amb l'usuari, que sigui senzill i entenedor, perquè ell mateix pugui reparar els possibles errors que hagin sorgit prèviament. Amb aquest objectiu es va crear el sistema EE4A que posteriorment esdevindrà la base del meu treball final de grau.

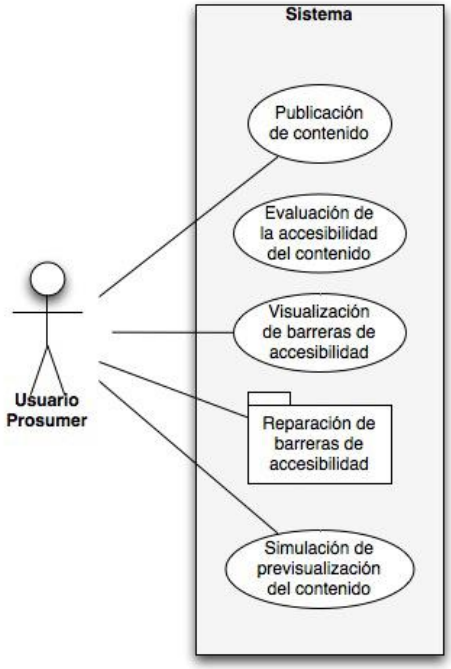


Il·lustració 6. Disseny del sistema EE4A

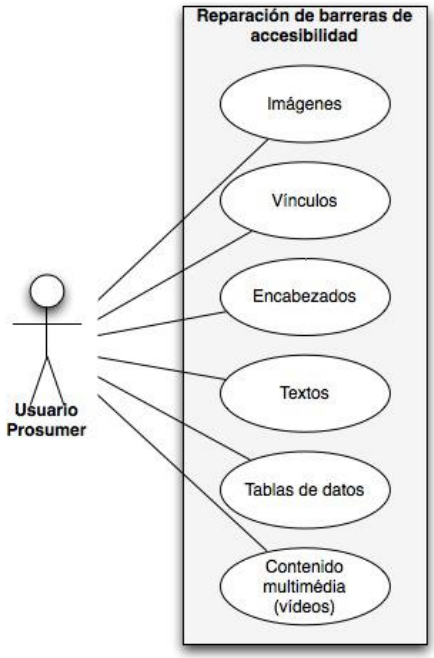
Els objectius específics que es van definir pel sistema EE4A eren els següents:

- Proposar un sistema que comuniqui les barreres d'accessibilitat d'una manera més empàtica, oferint l'usuari una perspectiva personal de les barreres de contingut i simulacions de discapacitats tal com ho pot percebre una persona amb discapacitat.
- Facilitar la reparació de les barreres d'accessibilitat, proporcionant instruccions concretes de les accions que ha de realitzar l'usuari per evitar el contingut no accessible.
- Presentar simulacions del contingut que ha escrit l'usuari, tal com ho percep una persona amb discapacitat.
- Generar un codi completament accessible dels elements web més rellevants, realitzant modificacions al codi font perquè al publicar-lo, compleixi amb els nivells d'accessibilitat adequats.

Amb aquests objectius es va iniciar la fase d'enginyeria de requeriments.



Il·lustració 8. Diagrama de casos d'ús general (EE4A)



Il·lustració 7. Diagrama de casos d'ús de reparació de barreres (EE4A)

Caso de uso: Reparación de barrera de accesibilidad: Imágenes **Actores:** Usuario prosumidor **Propósito**: Introducir los datos necesarios para hacer una imagen accesible **Resumen**: Se introduce el texto alternativo que describe la imagen **Tipo**: Primario y esencial **Excepciones**: No hay excepciones	
Acciones de los actores	**Respuesta del sistema**
	Paso 1: Se presentan datos relacionados con la imagen: • Tipo de imagen: se indica al usuario que revise la información relacionada con la imagen, y que elija si es: de texto/de decoración/de contenido.
Paso 2: El usuario selecciona el tipo de imagen	
	Paso 3: Introducción de datos. Según el tipo de imagen seleccionada por el usuario: • De texto, de contenido: se muestra un cuadro de texto para que el usuario añada un texto alternativo a la imagen (Se muestra el texto actual del elemento ALT. Se muestra sugerencia) • De decoración: No se muestra cuadro de texto.
Paso 4: El usuario introduce texto alternativo en la imagen	
	Paso 5: El sistema modifica el código fuente que corresponde a la imagen editada para que cumpla con el nivel de accesibilidad adecuado: • De texto, de contenido: se añade el texto introducido por el usuario en la etiqueta ALT de la imagen • De decoración: se añade un espacio a la etiqueta ALT

Il·lustració 9. Cas d'ús de reparació d'imatge EE4A

Com es pot apreciar en la il·lustració 9, ja s'esmenta com ha de ser aquest diàleg entre l'usuari i el sistema per a poder reparar les barreres. En aquest cas en concret, les modificacions que realitzaria el sistema serien les següents:

• Opció 1

• Opció 2

<img src="gat.jpg" alt="Text afegit per l'usuari en el quadre text"
[altres atributs HTML] />

• Opció 3

<img src="gat.jpg" alt=" Text afegit per l'usuari en el quadre text"
[altres atributs HTML] />

També es van definir com haurien de ser els formularis:

Texto informativo a mostrar al usuario para la reparación:
La imagen debe ir acompañada de un texto que la describa

Indica el tipo de imagen:

- ○ (opcion1) Es una imagen decorativa, sin importancia en el contenido
- ○ (opcion2) Es una imagen relevante en el contenido
- ○ (opcion3) Es una imagen de texto, ¿qué texto contiene?

Opción 1-- (no se hace nada). ALT vacío.
No debes hacer nada. Los usuarios ciegos no verán esta imagen y no es necesaria para entender el contenido.

Opción 2
Sugerencia → *Describe esta imagen cómo si la estuvieras explicando a alguien por teléfono*

Descripción de la imagen actual [gato]

Opción 3 (en este ejemplo no irá a esta opción, porque la imagen no es de texto)

Sugerencia → *Escribe el texto que se muestra en la imagen*
Descripción de la imagen actual [gato]

Il·lustració 10. Exemple de reparació d'imatges

Texto informativo a mostrar al usuario para la reparación:
Verifica que el texto del enlace es adecuado a su destino o funcionalidad con un texto conciso y significativo en el texto del enlace.

Se muestra la dirección URL donde vincula el enlace

- ○ URL del enlace: [`http://www.google.com`]

El texto del enlace ha de indicar adecuadamente su destino

- ○ Texto del enlace: [enlace] (sugerencia) → [Página de google (abrir en nueva ventana)*]
  *Se obtiene accediendo al título de la página de destino

El título del enlace se utiliza únicamente para ampliar información del texto del enlace

- ○ Título del enlace [ ] (sugerencia) → [Abrir página de google]
  *Si Indicar el tipo de enlace que se abrirá. (ejemplo Buscador)

Il·lustració 11. Exemple de reparació de vincles

Al final de tota la tesi s'arriba a definir com reparar Imatges, Vincles, Capçaleres, Textos, Taules de dades i vídeos. En aquest punt es comença a prototipar i implementar el sistema EE4A i les seves funcionalitats.

Al final de tot aquest treball, en el punt el qual arribo jo, em trobo el sistema EE4A completament funcional per a reparar vincles i imatges. En aquest punt faltaven per implementar diversos formularis de reparació i diverses funcionalitats extra. Tanmateix, el sistema estava pensat per a comunicar-se amb els CMS, per tant, em vaig disposar a iniciar el meu treball, implementant una API per a poder comunicar-me amb un Plug-in de WordPress.

3. Gestió del Projecte

Per a la realització d'aquest projecte es van definir un conjunt de fites i una llista de tasques per assolir-les simulant una metodologia de desenvolupament àgil. En el moment d'iniciar el projecte es van definir les següents fites:

1. Anàlisis del material obtingut
    a. Anàlisi del treball anterior
    b. Anàlisis de disseny BBDD
    c. Anàlisi de disseny Symfony
2. Implementació API Servidor
3. Plug-in
    a. Disseny
    b. Implementació
    c. API Client
    d. Testing
4. Afegir funcionalitats
5. Documentació
6. Publicació

La primera fase del projecte consistia en l'anàlisi. Bàsicament la primera part consistiria en estudiar i comprendre com s'havia construït l'aplicació EE4A. L'objectiu era comprendre que era, quins objectius es plantejava, quina estratègia seguia, com s'havia estructurat, entendre el funcionament, i com i on s'havia de construir l'API. A més aquesta fase incloïa el procés d'entendre el treball anterior al projecte, d'on havia sorgit i en el punt en el qual es trobava. En aquesta fase inicial també hi formava part la instal·lació de l'aplicació en un entorn local, per a poder començar el desenvolupament en la següent secció correctament.

La fase d'implementació de l'API en el servidor bàsicament consistia en programació en llenguatge PHP utilitzant les llibreries adequades, per a poder implementar una API amb la qual comunicar-nos amb el servidor.

La secció del Plug-in és la que té més pes i treball en desenvolupament del projecte. Bàsicament consta de la implementació de l'aplicació AngularJS que ens servirà com a extensió en l'entorn CMS WordPress. Aquesta fase va des del disseny inicial fins a la implementació de la comunicació entre el servidor i aquest. Respecte a la documentació, aquesta està realitzada en el codi, on els controladors estan documentats correctament. La resta de documentació més de referència és la que es pot trobar en aquest document.

Al cap d'unes setmanes de treball van anar sorgint nous requeriments que no estaven previstos en la llista de tasques. En aquest cas es va haver d'afegir la tasca d'Integració del Plug-in dins de WordPress. Aquesta nova tasca consistiria a integrar l'aplicació dins de l'entorn, ja que s'han d'utilitzar diverses guies i funcionalitats internes de l'entorn CMS. En un principi no estava comptada com a tasca, ja que es pensava que seria breu i senzilla. Finalment aquesta tasca va resultar una mica més complicada del que s'esperava, per aquest motiu es va haver d'afegir a la llista de tasques posteriorment.

A totes aquestes fases s'hi va donar un pes, en funció del cost temporal que estimava que es tardaria a realitzar. L'orde de realització en un principi estava plantejat en un ordre lògic, en aquest cas em refereixo a què primer es realitzava l'anàlisi, posteriorment la part del servidor i posteriorment el client. Tanmateix degut als problemes que van sorgir durant el desenvolupament, l'ordre va anar variant en funció dels problemes o dels canvis aliens al desenvolupament del projecte.

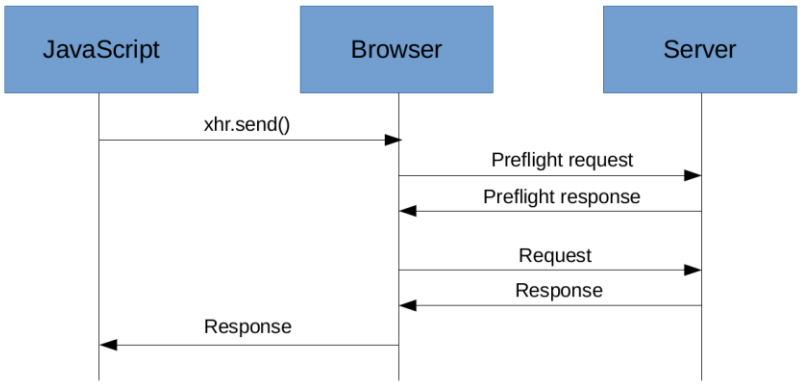
Un dels primers problemes que va sorgir va ser a causa de l'Achecker. Al començar a desenvolupar d'una base ja existent, aquesta no té per què funcionar perfectament. Aquest va ser el cas, ja que després d'unes setmanes de retard a causa de dificultats en la instal·lació inicial, després de realitzar diverses proves vaig descobrir que hi havia un error originat en l'Achecker.

L'error en qüestió era originat pels accents, els quals produïen que les avaluacions no es realitzessin correctament pels següents motius. En primer lloc, perquè no podia començar la implementació de l'API del servidor, sabent que hi havia un error en el servidor. En segon lloc, no podria comprovar amb seguretat que l'aplicació estigués funcionant correctament. En tercer lloc i finalment, tampoc podia estar segur de què no tingués que desfer feina, es va decidir posposar la tasca d'implementació de l'API.

En aquest punt es va començar a crear l'estructura per a poder incloure l'aplicació en el WordPress, és a dir, es va iniciar parcialment la fase d'integració. Un cop ja es tenia la base sobre la qual començar es va iniciar la fase de desenvolupament de l'aplicació AngularJS, variant així l'ordre previst inicialment, com es pot comprovar en la secció de temporalitat final.

Durant la resta del desenvolupament del projecte també s'han produït diversos contratemps o problemes menors que en cert moment han pogut interrompre el desenvolupament del projecte. Un exemple seria la que es va realitzar durant el desenvolupament de l'API en l'aplicació AngularJS. Després d'haver provat l'API via directa (Postman i similars) no hi havia cap problema, però a l'hora d'executar-la dins del navegador van començar a sorgir errors. En aquest cas eren errors relacionats amb el CORS.

El CORS és un sistema de seguretat per impedir que es realitzin peticions HTTP a recursos externs al domini d'una pàgina iniciats des d'un script.



Il·lustració 12. Esquema de funcionament CORS

Bàsicament el navegador realitza una request amb la capçalera Access-Control-Allow-Origin amb el valor del domini d'on surt, i si el servidor l'accepta, es permet la connexió. A més, en alguns casos es realitza un Preflight consisteix en una consulta prèvia on es comprova que el mètode està permès i si ja no l'accepta, la petició ja ni es realitza. Tot el que he explicat està molt més estès en les següents pàgines (CORS, 2016), (Access Control: CORS, 2016).

Aquest problema va aturar el projecte uns quants dies fins que vaig prendre la decisió de seguir desenvolupant el client i deixar la connectivitat pel final, perquè algunes crides a l'API sí que funcionaven correctament, però el projecte no es podia endarrerir.

4. Desenvolupament

4.1 Primera Part: Fase d'anàlisi

4.1.1 Anàlisi treball anterior

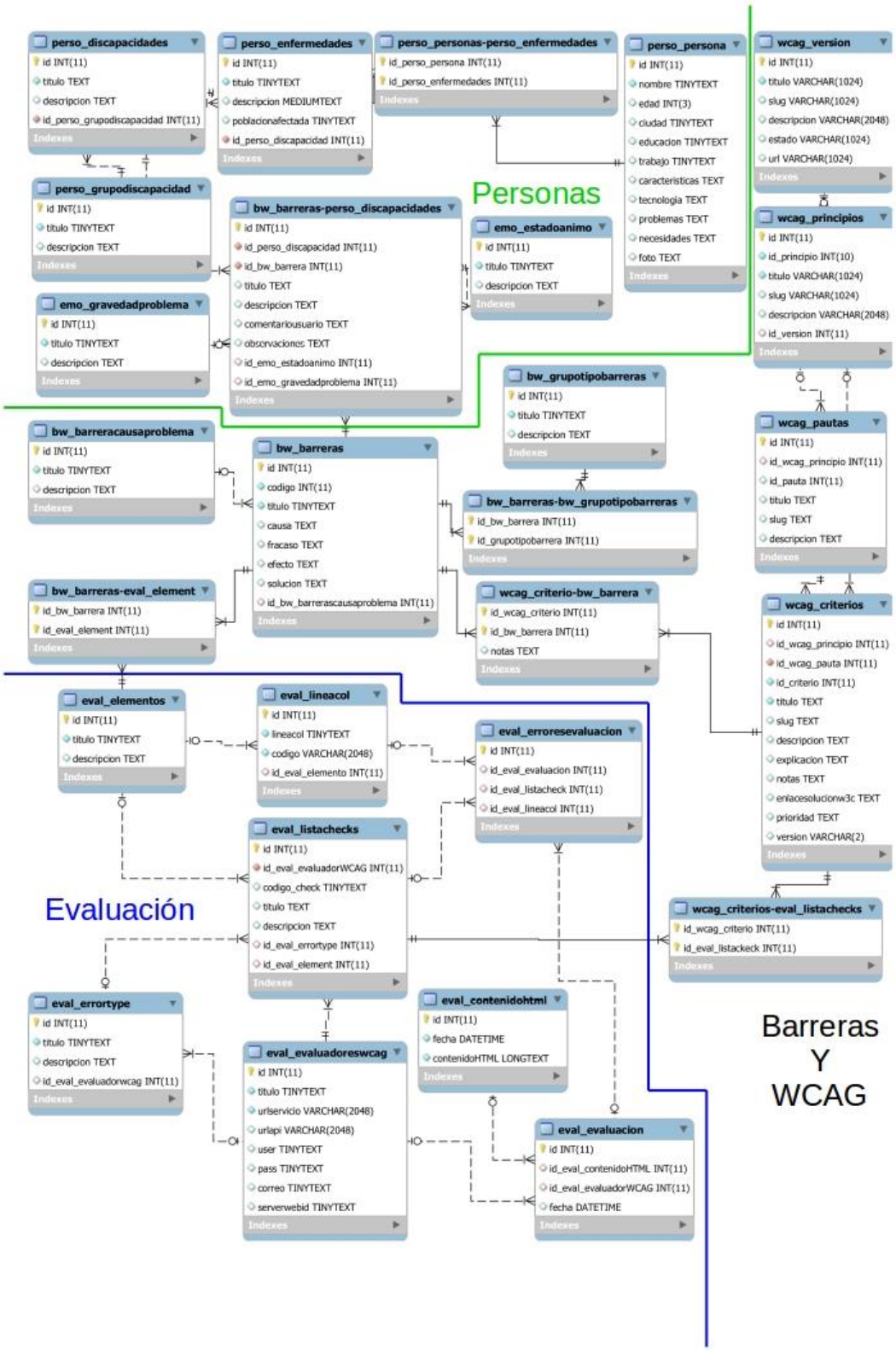
En aquest projecte de final de grau es va tenir molt present tot el treball anterior al punt de partida. Aquesta primera fase consistia a analitzar tot el material obtingut per poder comprendre el treball previ que hi havia anterior al meu projecte, el disseny i funcionament de l'aplicació "Emphatic Editor for Accessibility". Tota aquesta informació es pot consultar en el punt 3.2

Després d'analitzar i entendre l'objectiu del projecte vaig tenir accés al repositori, i a continuació, vaig procedir a descarregar-me i instal·lar-me tant l'aplicació, com la base de dades, en un entorn local. En aquestes primeres setmanes, tal com estava planificat em vaig disposar a analitzar el codi de l'aplicació web, programada amb PHP Symfony2, i la Base de Dades amb la que treballava.

4.1.2 Anàlisi BBDD

Tal com s'havia planificat es va començar amb l'anàlisi de l'estructura de la Base de Dades. Una base de dades d'aquestes característiques no és fàcil de fer ni d'estudiar, ja que està composta per múltiples taules enllaçades.

En aquest punt realitzar un estudi centrant-nos únicament amb el disseny pot ser pràcticament impossible. Per tant la solució va ser anar seguint el flux del codi i al mateix temps anar comprovant en quines accions intervenia la base de dades i quina funció tenia.



Il·lustració 13. Estructura de la base de dades E4A

En aquest cas podem dir que la base de dades està dividida en tres seccions fonamentals. La primera part està marcada com a Evaluación, de color blau, en la il·lustració 13. Aquesta part s'inicia en "eval_contenidohtml" que és la taula on s'emmagatzema el contingut HTML avaluat. Al mateix temps que s'emmagatzema es crea l'avaluació que en el nostre cas només es realitza amb el Achecker. El procés d'avaluació es realitza via API contra l'Achecker que també es troba en local. Un cop aquest ens retorna el resultat s'introdueix en les següents taules de la base de dades amb prefix "eval".

Per altra banda hi tenim les Barreres i WCAG, de color negre, en la il·lustració 13. Un cop realitzada tota l'avaluació, aquesta es relaciona amb les barreres d'accessibilitat que ja tenim definides en la base de dades. Al mateix temps, també relacionem aquestes amb els criteris WCAG, que també tenim en la base de dades. Tot aquest procés es realitza per a poder informar sobre les barreres i per identificar com poder solucionar aquestes.

Finalment en la secció superior de la il·lustració 13 ens trobem les taules de la zona Personas, de color verd, amb el prefix "perso". Aquestes taules tenen la funció d'emmagatzemar la informació de persones fictícies. L'objectiu és donar nom a les persones que han de conviure amb les barreres d'accessibilitat web diàriament.

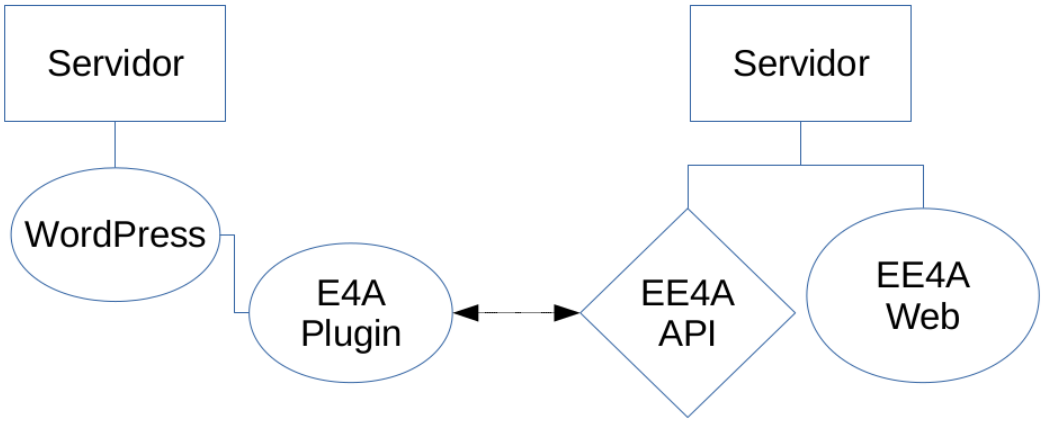
4.1.3 Anàlisi Symfony

Un cop entès l'objectiu i funcionament de la base de dades va ser el moment d'endinsar-se dins del comportament i flux del codi. La complexa estructura de Symfony, la qual està composta per seccions anomenades "Bundles", fa que sigui bastant més complicat entendre el funcionament del codi a primera vista.

Més que identificar els components i com realitzaven les funcionalitats, em vaig centrar a estudiar el comportament de l'aplicació web, totes les seves vistes, ja que posteriorment hauria de realitzar, un Plug-in que s'assemblés el màxim possible a l'aplicació existent. A més a més també calia estudiar el lloc on crear un nou "Bundle" que funcionés com a API Rest i definir aquesta.

4.2 Segona Part: Fase de desenvolupament

Clarament es poden diferenciar les dues seccions de desenvolupament en aquest treball. Per una banda, a la dreta de la il·lustració 14, es troba la part del servidor en Symfony, en el qual s'ha de realitzar una API Rest, i per altra banda, a l'esquerra, ens trobem en què es realitzarà una aplicació web en AngularJS la qual ens servirà de Plug-in en el WordPress.



Il·lustració 14. Esquema global del Projecte

4.2.1 Servidor

El servidor, segurament el que semblava la tasca més fàcil i una de les que estava planificada en els inicis del projecte amb un pes molt baix. Probablement, ha sigut la tasca més complexa i amb més contratemps de tot el treball. Pot semblar que no, però es tracta d'un punt crític, si no és possible la comunicació entre el servidor i el client, el treball en si no pot seguir endavant. Centrant-me únicament en l'aspecte tecnològic explicaré el desenvolupament d'aquesta part.

Inicialment es va crear el controlador de "Personas". Aquest al ser el controlador més simple em va permetre poder-lo utilitzar com a base per a les demes accions més complexes. L'objectiu principal és que donat un identificador et retorni tota la

informació d'una de les persones definides en l'aplicació EE4A en format JSON, o que les retorni totes si no s'introdueix cap paràmetre.

Posteriorment, va començar la implementació del controlador "Eval". Aquest controlador tindria la funció de rebre el HTML a avaluar i generar una avaluació. Tot semblava anar bé, però van sorgir diversos problemes amb el mètode POST. En aquest punt van sorgir problemes relacionats amb els CORS, els quals ja han set esmentats en l'apartat de Gestió del Projecte.

Un cop solucionats tots els problemes vaig poder prosseguir amb el desenvolupament del servidor. En aquest moment només es necessitava un últim controlador seria l'encarregat de gestionar les reparacions.

Les reparacions de contingut, els formularis els quals ha d'emplenar l'usuari, es troben en el servidor, juntament amb tota la lògica. Per tant l'única manera d'aconseguir accedir-hi des del client és enviant el formulari amb HTML via API. Un cop s'ha obtingut el formulari s'ha de mostrar en pantalla perquè l'usuari el respongui i es pugui enviar de tornada cap a l'aplicació EE4A, la qual aplicarà tots els procediments que s'hagin de realitzar.

Finalment, un cop acabades les operacions, quan es prem el botó de finalitzar en el client, es fa una crida a Evals on ens retorna el HTML final arreglat. Un cop aquest és obtingut solament ha de ser substituït per l'HTML original.

A part d'aquests mètodes n'hi ha un el qual ens retorna tota la informació necessària per poder veure informació sobre una persona en concret, per la vista de les persones en el client.

Així doncs, finalment l'API va quedar definida de la següent manera:

Mètode	**HTTP Request**	**Descripció**
getPersonas	GET /api/personas/{:id}	Retorna tota la informació de les persones definides en l'aplicació o només retorna la indicada el id opcional, en format JSON

getEval	GET /api/evals/id	Retorna el resum de l'avaluació EE4A en format JSON
postEval	POST /api/evals Format {'html': 'value'}	Retorna el id de resum generat en format JSON Format {'id': 'value'}
getEvalHtml	GET /api/evals/id/html	Retorna el HTML modificat després del procés en format JSON
getReparar	GET /api/reparar/idResumen/idDiscapacidad/idBarrera/id	Retorna el formulari HTML a emplenar per l'usuari
postReparar	POST /api/reparar/idResumen/idDiscapacidad/idBarrera/id	Retorna 200 si ha funcionat correctament

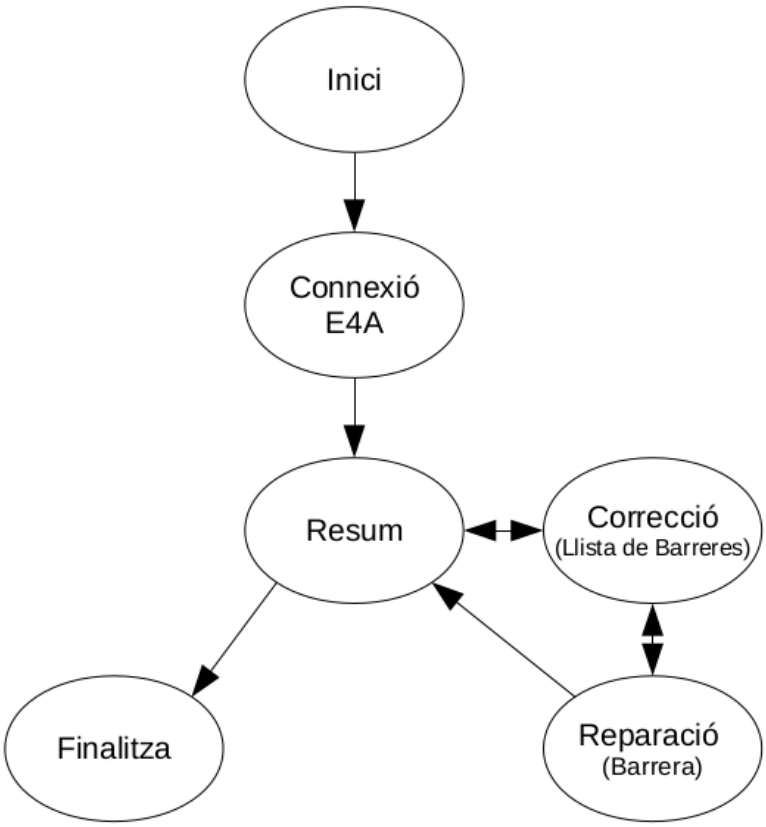
4.2.2 Client

Funcionament conceptual

La part la qual s'ha emportat més pes i més treball en aquest projecte ha estat el client, ja que és la part amb la qual l'usuari interactuarà dins del WordPress.

La primera fase va consistir a dissenyar quin havia de ser el funcionament de l'aplicació. En estar executant-se dins de l'entorn WordPress no podia canviar la direcció URL i alhora tampoc podia guardar formularis de manera habitual, ja que recarregava la pàgina del WordPress i es perdia tot el progrés. Per tant s'havia de realitzar una aplicació amb múltiples vistes i funcionalitats, la qual pogués conviure amb l'entorn WordPress i que alhora no canvies de ruta URL.

Per evitar totes les URL es va plantejar i dissenyar un diagrama d'estats. Amb el framework AngularJS podem treballar únicament amb estats evitant així les URL. La màquina d'estats amb la qual funcionaria l'aplicació seria la següent:



Il·lustració 15. Diagrama d'estats del Plug-in

L'estat inicial de l'aplicació ("Inici" en la il·lustració 15) s'inicia en el moment de prémer el botó de EE4A al menú d'administració de WordPress. En aquest moment s'obre un "modal dialog" el qual conté tota l'aplicació client. En aquest estat únicament es pregunta a l'usuari si vol realitzar la revisió d'accessibilitat i en cas afirmatiu es passa al següent estat. Si en canvi l'usuari declina, l'aplicació finalitza.

En cas que l'usuari vulgui realitzar l'avaluació es passa al segon estat: Connexió EE4A. La funcionalitat d'aquest estat és connectar-se a l'API implementada en el servidor per enviar-li el codi HTML a avaluar, que és el que l'aplicació ha llegit de l'editor del WordPress, i un cop enviada aquesta informació ha d'obtenir el resum en format JSON per poder crear les vistes que vindran a continuació.

Suposant que el procés ha funcionat correctament, s'arriba a l'estat Resum. L'estat resum ens mostra les diferents persones, cadascuna associada a la seva discapacitat, i ens informa del nombre de barreres afectades, si es publica aquest contingut. En aquest moment l'usuari pot decidir si les vol reparar, o si en canvi desitja finalitzar i acabar el procés.

En cas que l'usuari desitgi reparar les barreres, accedirà primer a l'estat Correcció. L'estat Correcció ens mostra les diferents barreres en llista, una breu descripció d'aquestes juntament amb la seva gravetat. En aquest punt l'usuari pot decidir quina barrera reparar, o bé, tornar enrere al resum.

Un cop decidida la barrera a reparar, entra a l'estat Reparació. Aquest estat ens mostra un formulari el qual l'usuari emplenarà per tal de corregir la barrera. Un cop enviat el formulari al servidor l'usuari podrà tornar al resum on veurà totes les barreres que queden per reparar, i podrà decidir si arreglar-les o no.

Finalment, quan l'usuari decideix acabar amb l'avaluació entra en joc l'últim estat: Finalitza. En aquest punt, l'aplicació es connecta contra el servidor per rebre el codi HTML arreglat. Un cop ha rebut el codi, substitueix el HTML de l'editor WordPress per aquest i finalitza l'aplicació.

Un cop acabat l'anàlisi del diagrama d'estats és el moment d'avaluar com realment funciona el Plug-in i com ha estat estructurat de manera que totes les peces encaixin.

El Plug-in s'ha construït respectant el patró MVC i intentant que el codi d'aquest sigui fàcilment adaptable i extensible per a millores o ampliacions de funcionalitats. Per generar un esquelet base d'on poder partir es va fer servir el generador AngularJS de Yeoman, el qual construeix una base sobre la qual començar a treballar.

En aquest punt del desenvolupament ens trobem amb què tenim l'estructura del Plug-in preparada per començar a treballar. En aquest moment era l'hora de crear el Plug-in dins de l'entorn WordPress.

Integració en l'entorn WordPress

Per procedir a integrar-lo, el primer que vaig haver de fer va ser descarregar-me l'última versió del gestor de continguts WordPress. Un cop descarregat i configurat correctament s'han de seguir unes pautes, les quals estan molt ben definides en la pàgina web, ja que WordPress és un entorn enfocat a poder rebre gran quantitat d'extensions.

En un primer moment es volia intentar que el Plug-in entrés en acció en el moment de publicar un post. A l'hora de fer clic saltaria el modal per poder realitzar l'avaluació. Aquest era l'objectiu inicial, però per qüestions temporals, no es va poder realitzar. Com a solució alternativa es va decidir incorporar un botó en l'editor de posts el qual en ser clicat llença l'aplicació i realitza tot el procés.

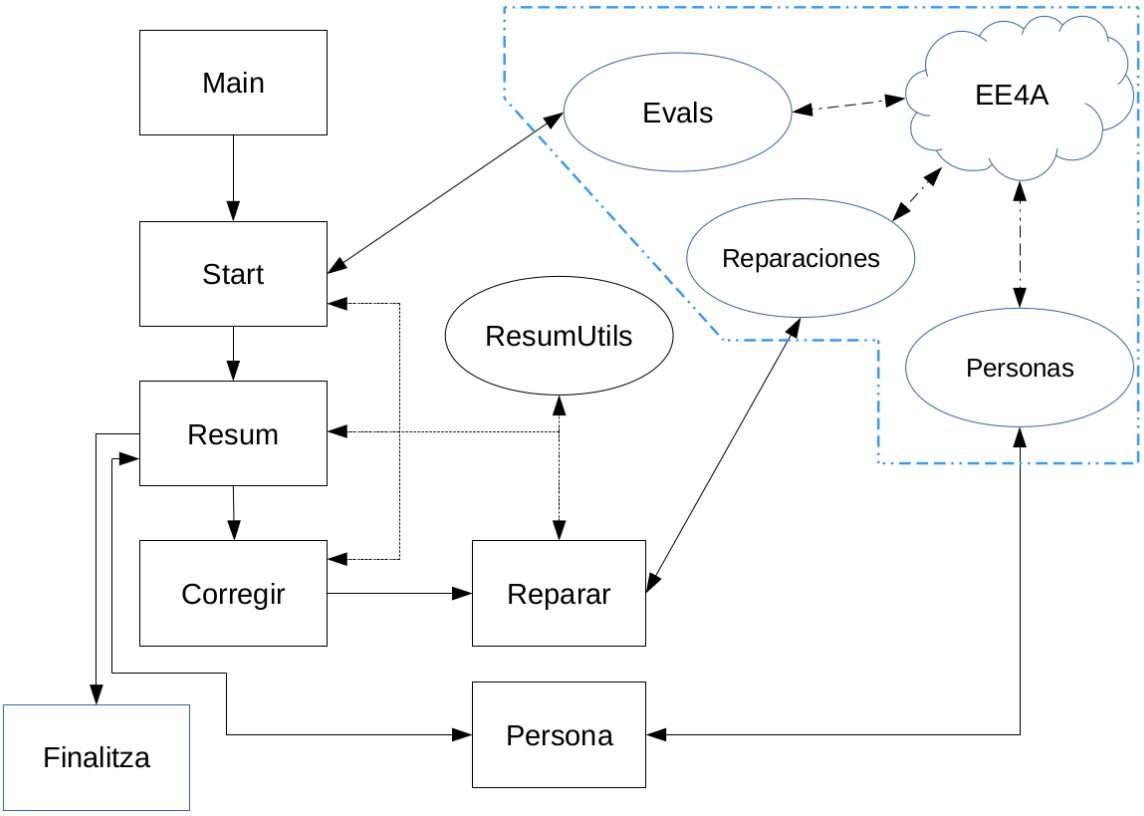
Il·lustració 16. Botó E4A integrat en el WordPress

A més de l'esmentat anteriorment, es va crear una pàgina d'administració del Plug-in en la secció d'opcions del WordPress, la qual pot ser de gran utilitat en un futur alhora de realitzar la implementació de la possible autentificació

Il·lustració 17. E4A en el administrador de Plug-ins

Estructura de l'aplicació client

Un cop feta la primera part de la integració del Plug-in dins de l'entorn WordPress, es va iniciar la fase de desenvolupament de l'aplicació en AngularJS. A continuació es mostra el diagrama final amb tots els components, sense incloure les vistes, ja que són homònimes respecte als controladors.



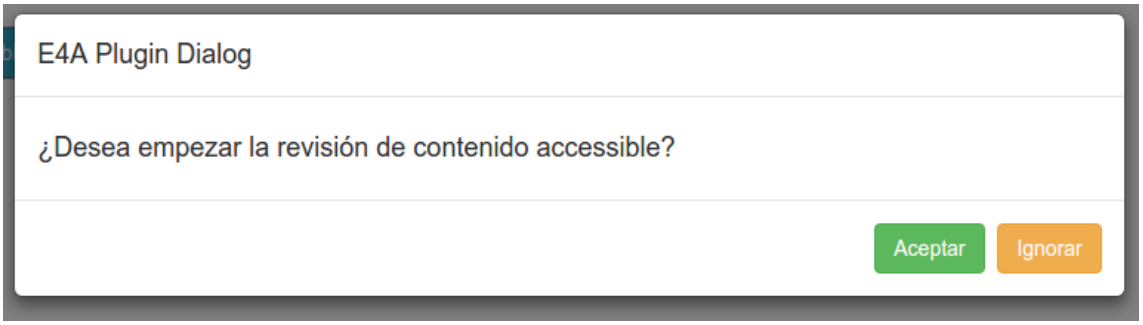
Il·lustració 18. Diagrama de components del client

Com es pot apreciar clarament en la il·lustració 18 estan definides tres seccions:

- La secció superior dreta, la qual està marcada amb color blau, és la secció orientada a la comunicació amb l'API. Evals, Reparaciones i Personas funcionen com a intermediàries entre el Plug-in i EE4A.
- La secció oposada, a l'esquerra, ens trobem tots els controladors de l'aplicació. Aquests implementen la lògica del Plug-in i tenen una vista associada amb el mateix nom.
- En la part central trobem ResumUtils. Aquest component és una factoria interna que conté tota la informació de l'avaluació. Tots els controladors hi poden llegir i escriure informació. Ara entrem a parlar més detalladament sobre els components i que realitzen exactament.

Disseny i interfície d'usuari

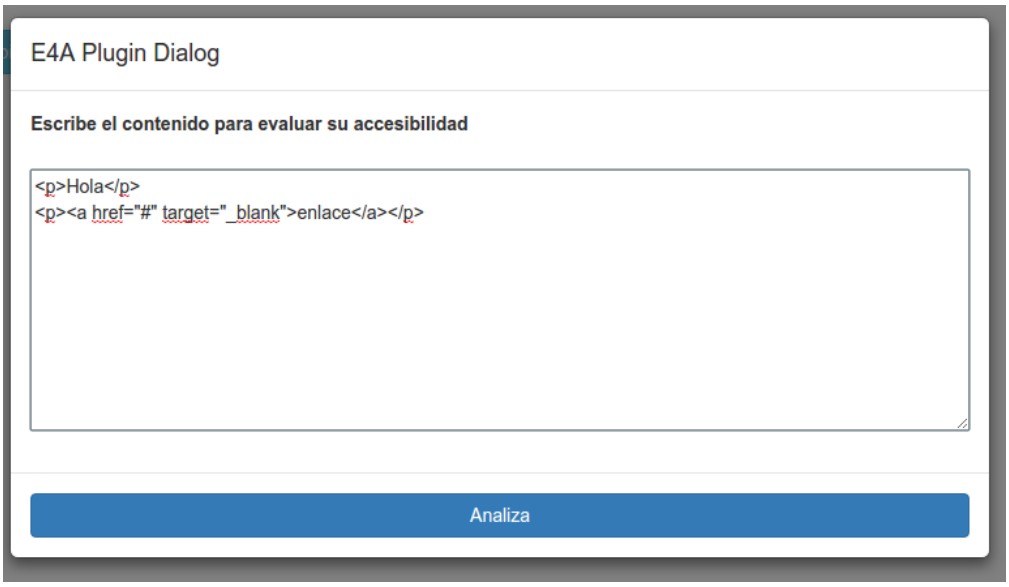
El primer en entrar en acció és el Main. És el controlador inicial de l'aplicació, el punt de partida, la seva lògica és molt simple, ja que es tracta d'una pantalla de confirmació per poder confirmar que l'usuari desitja començar l'anàlisi o en canvi no el vol realitzar. En ser un modal en cas de no voler realitzar l'avaluació, aquest desapareix.



Il·lustració 19. Vista inicial

En cas que l'usuari desitgi continuar amb l'avaluació de contingut, l'estat canvia i entra en acció el controlador Start. Aquest controlador té dos objectius principals:

- Per una banda, el primer objectiu consisteix en obtenir una confirmació visual del contingut, i d'aquesta manera ens assegurem de què estem avaluant el contingut correcte. Si l'usuari el verifica, aquest controlador inicia el procés de comunicació amb EE4A.



Il·lustració 20. Vista de comprovació

- El segon objectiu es produeix un cop iniciat aquest procés de comunicació, on es visualitza una pàgina de càrrega la qual ens mostra el progrés en les comunicacions amb el servidor i construcció del resum. En un principi, el que fa és cridar al component Evals que realitza una petició POST al servidor amb el codi HTML a avaluar. En aquest moment, el controlador s'espera a obtenir la resposta que serà l'identificador de l'avaluació. En el supòsit que tot funcioni correctament, realitza una crida GET a través de la mateixa factoria (Evals), la qual ens retorna un gran JSON amb tot el necessari per a poder construir el resum en el client.

Il·lustració 21. Vista de carrega correcta

Tanmateix, encara queda una part important del procés. Un cop s'ha rebut el JSON l'envia a ResumUtils i crida la funció "*doResum()*". Aquesta funció és essencial per poder construir el resum en el client, ja que converteix el JSON amb objectes diferents que després seran de gran utilitat per poder generar totes les vistes automàticament. A més, també es comptabilitza el nombre d'errors, perquè no bé definit en l'aplicació.

La decisió d'utilitzar ResumUtils com a "centre de dades" de tot el client va sorgir a causa de la necessitat de generar múltiples vistes depenent únicament del JSON obtingut. A més, en tenir tota la informació tractada en un punt és més fàcil tenir controlats els possibles errors en aquest aspecte, ja que si anéssim enviant la informació de controlador en controlador com a paràmetre, lògicament, seria molt més complicat localitzar l'error.

En relació al controlador *Start* anteriorment mencionat, un cop tot el procés ha finalitzat ens dóna l'opció de continuar cap al Resum. No ha estat comentat abans, però òbviament, en el procés d'enviament de dades pot sorgir algun tipus d'error. Si es produeix algun error de connexió o un error en la generació del resum es mostra una pantalla d'error la qual ens permet tornar enrere a la pantalla de validació i un cop allà es pot tornar a intentar la connexió.

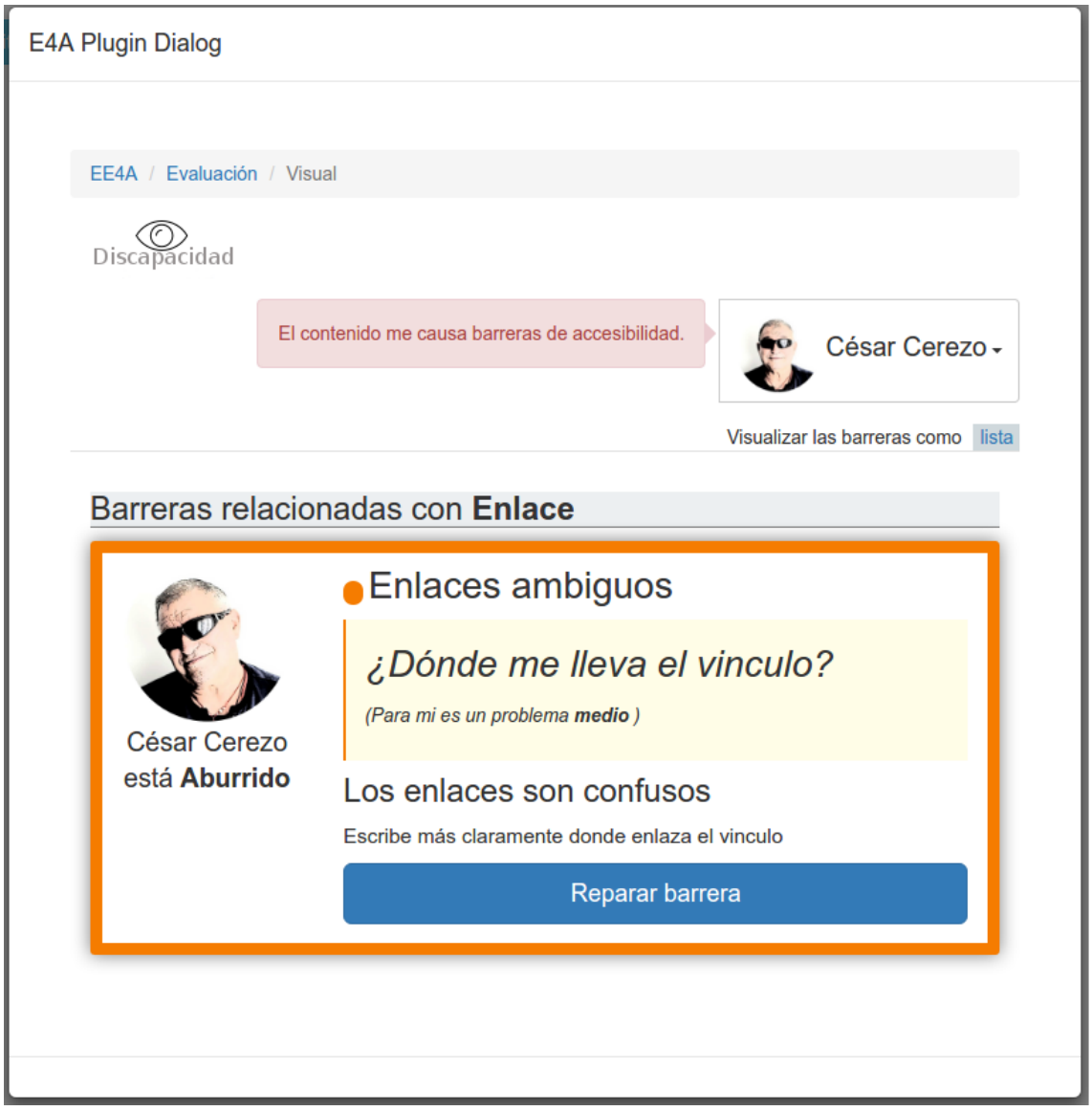
Il·lustració 22. Vista de carrega (amb error)

Un cop hem passat tota la fase inicial, s'arriba al centre de l'aplicació: el resum. En aquesta vista se'ns mostren les discapacitats avaluades, el nombre de barreres afectades alhora que se'ns mostra el nom de les persones.

Il·lustració 23. Vista resum

Un com hem arribat a aquest punt, podem apreciar les barreres que estan afectades i tenim l'opció de corregir-les. En cas de prémer el botó finalitzar, l'aplicació es dirigirà a l'estat finalitza i acabarà l'avaluació.

En cas que l'usuari vulgui corregir, com he dit anteriorment, passem a l'estat Corregir. En aquest estat se'ns mostra una llista de les barreres afectades ordenades per grups.



Il·lustració 24. Vista corregir

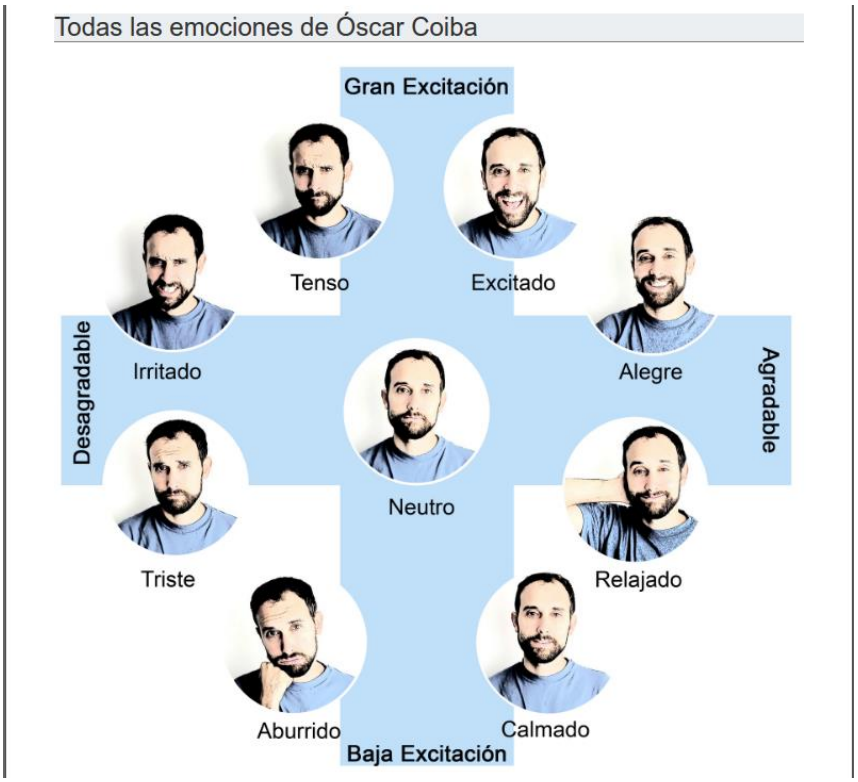
En aquest punt l'usuari visualitza les barreres afectades. Un cop aquí es pot entrar a reparar una barrera o tornar enrere al resum. A més, en l'apartat on es troba la persona podem prémer una de les opcions i ens redirecciona a la vista de persona la qual ens permet obtenir informació d'aquesta.

Il·lustració 25. Vista persona (primera part)

La vista persona està dividida en dues seccions:

- La primera secció (la il·lustració 25) correspon a la part superior de la pantalla. Un cop entrat en la vista persona, podem visualitzar tota la informació d'aquesta. L'objectiu d'aquestes persones, tal com està definit en la tesi, és poder emfatitzar amb elles i reparar les barreres d'accessibilitat correctament.

- La segona secció, part de la vista consisteix en una gran imatge on es defineixen totes les possibles emocions d'aquestes persones. Aquestes emocions apareixeran en el sistema en funció del nombre de barreres que afectin aquestes persones. En la il·lustració següent es pot veure la imatge.



Il·lustració 26. Vista persona (segona part)

Tornant enrere, a la vista de corregir errors, també pot ser que a l'hora de voler reparar una barrera, pot ser que el formulari de reparació no estigui implementat en el servidor. En aquest cas apareixerà una alerta informant-nos que la funcionalitat no està disponible.

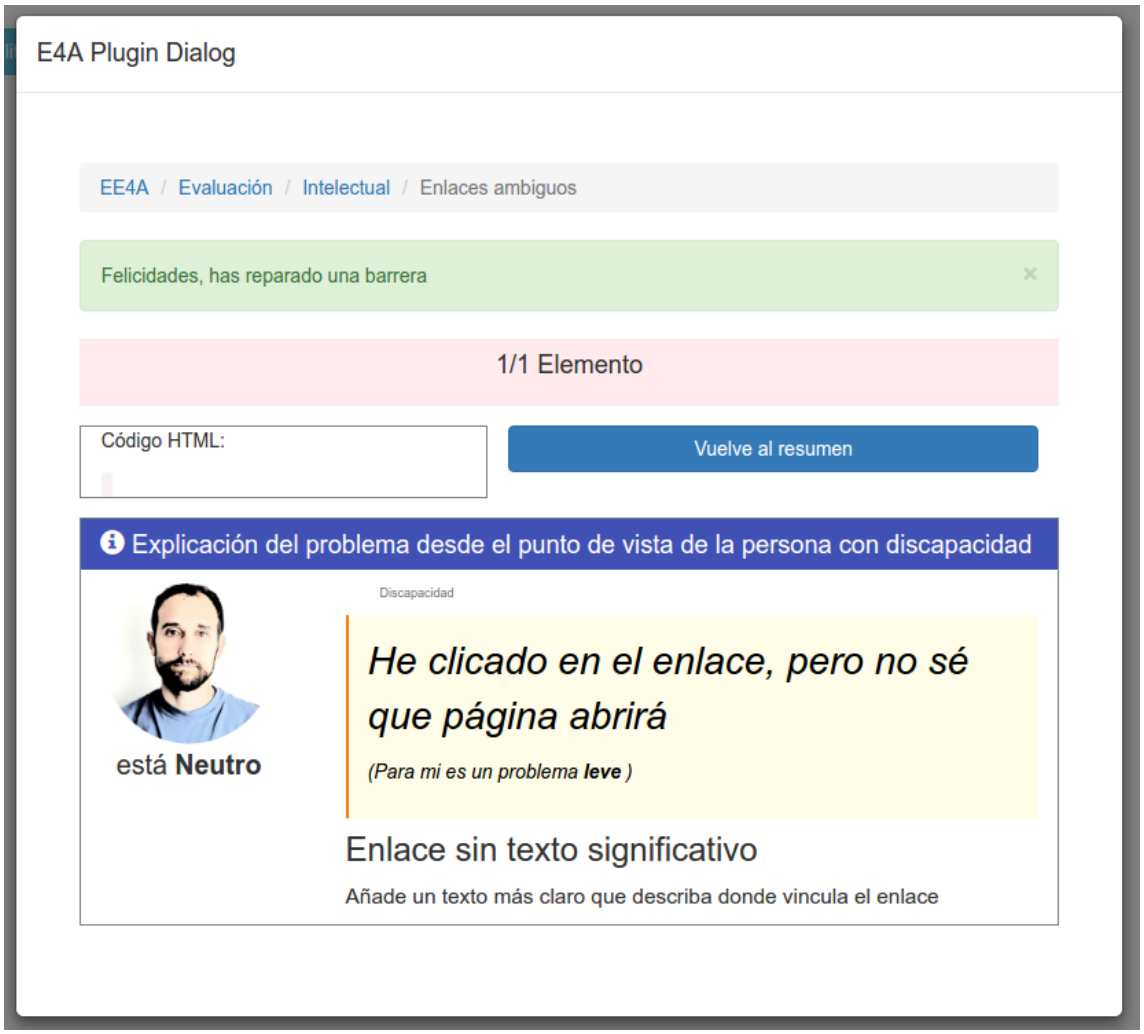
Il·lustració 27. Error a l'hora de reparar

Si per altra banda tot funciona correctament, ens dirigirem al controlador Reparar.

Il·lustració 28. Vista reparar

Un cop arribats a aquest punt, ens trobem amb un formulari a emplenar per poder reparar la barrera. A més, podem tornar a veure la informació sobre la barrera en la part inferior. Un cop respost el formulari, obtingut via JQuery i a continuació, s'envia a través de la factoria Reparaciones cap a EE4A via POST.

Si no sorgeix cap tipus d'error durant el procés, ens sortirà una alerta indicant-nos que la barrera ha estat reparada correctament i ens apareix un botó per retornar al resum.

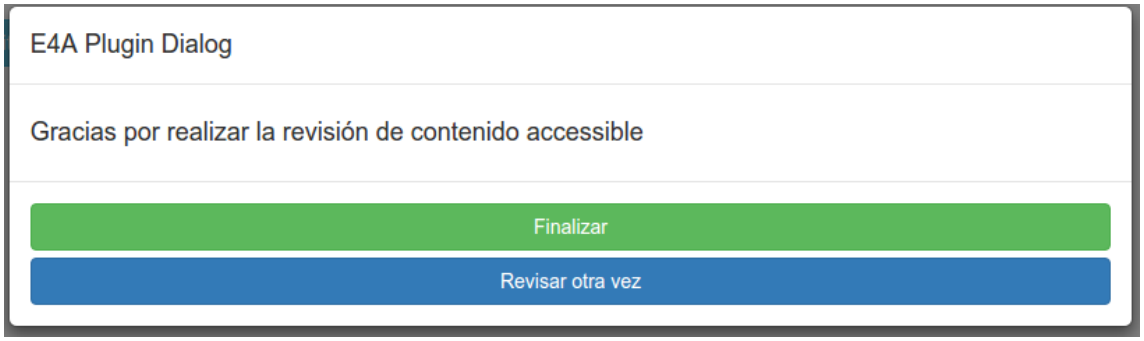


Il·lustració 29. Vista reparació correcta

En el moment que el formulari és acceptat pel servidor, el client interpreta que el problema ha estat resolt, per tant, en comptes de tornar a realitzar una crida GET i tornar a generar el resum actualitzat, en estar tot el resum en ResumUtils, simplement, el que realitza el client és eliminar la branca de l'arbre d'errors. Un cop realitzat, tot els altres controladors estan actualitats, ja que tota la informació del resum està centralitzada, per tant, no ens hem de preocupar més per la barrera.

Un cop hem solucionat totes les barreres, podem passar a finalitzar el procés. Un cop es decideix finalitzar no es pot tornar enrere sense iniciar de nou tot el procés. En aquest punt es crida al controlador Evals que ens retorna el HTML final arreglat del servidor. Un cop el tenim el HTML nou el substituïm per l'original i arribem a la vista final.

En aquest punt ens apareix un missatge de gràcies per haver realitzat l'avaluació i ens dóna l'opció de fer desaparèixer el modal o de tornar a iniciar l'avaluació.



Il·lustració 30. Vista final

D'aquesta manera, ha finalitzat el funcionament del client. L'aplicació client es minimitza perquè es pugui continuar la interacció amb el WordPress de manera normal. Si l'usuari torna a prémer el botó EE4A tornarà a la vista final, però podrà tornar a iniciar l'avaluació amb el nou contingut.

5. Conclusions

Personalment, crec que aquest projecte m'ha permès ampliar els meus coneixements tan pràctics com teòrics, ja que he hagut d'aprendre noves tecnologies i nous frameworks amb anteriorment mai havia treballat. Un altre dels aspectes enriquidors en la meva experiència, ha estat el fet d'haver de continuar un projecte, i el fet d'haver d'entendre tot el treball realitzat anteriorment.

El fet de partir d'un treball existent, en un principi pot semblar que facilita molt l'execució del present treball final de grau, ja que disposem d'un bon material inicial, però, com m'ha demostrat la practica, això no és així. Començar un treball, a partir d'una base implica conèixer aquesta a la perfecció per tal de poder modificar-la, ampliar-la o usar-la sense cap tipus de dificultats. En aquest treball, en concret, es va haver de comprendre el codi, les estructures, els fluxes i les funcionalitats existents per poder desenvolupar i executar la meva part correctament.

També, m'ha suposat un repte la situació d'haver de treballar amb múltiples tecnologies amb continguts diferents, per exemple, per una banda, estava el sistema EE4A amb Symfony; i, per l'altra banda, les pautes PHP del WordPress; sense oblidar l'AngularJS per desenvolupar tot el client.

Segurament, un dels punts més complicats alhora de realitzar el present treball, ha estat la temporalitat. Es complicat realitzar una previsió acurada d'un projecte amb aquestes dimensions. La realització acurada de la temporalitat, potser és una cosa que només es pot millorar amb experiència professional. En aquest cas, es va produir un desajust d'un mes de previsió aproximadament.

Un aspecte que m'agradaria reflexionar, per concloure, és la manera com s'han tractat els problemes per poder-los solucionar. Durant el desenvolupament del projecte s'han produït diversos contratemps o problemes menors que en cert moment han pogut interrompre el correcte desenvolupament del projecte. En aquest cas, el procediment era primer intentar solucionar el problema inicial, i si no era possible, sigui per falta de coneixement o per no aturar el projecte, vaig optar per passar al desenvolupament d'una altra tasca. Un cop desenvolupada l'altra tasca es tornava a intentar solucionar el

problema, el qual en la majoria de vegades, ja havia descobert com s'originava i com poder donar-li una solució acurada.

En ser un treball individual, si apareixen problemes o el projecte es queda encallat en un punt, i al tenir una temporalitat relativament fixa, no pot quedar aturat el projecte indefinidament en un punt, sinó que s'ha de seguir avançant en un altre punt – sempre que les circumstancies ho permetin- fins trobar la solució als problemes anteriors.

Respecte a les parts previstes que no s'han pogut realitzar per falta de temps, voldria esmentar-les a continuació:

- En primer lloc, la fase d'afegir funcionalitats, ja que l'aplicació del servidor requereix una re-estructuració llarga i complexa. Perquè s'implementen grans funcionalitats encara serà més complicada i personalment crec que és important arreglar primer el que està fet, abans d'endinsar-nos en aquells aspectes més complexes i que requereixen, com deia anteriorment, una labor molt més acurada i temporalment més extensa.

- En segon lloc, lògicament, la fase de publicació tampoc es va realitzar, perquè encara hi ha molta feina prèvia que realitzar, abans d'endinsar-nos amb la publicació final. Bàsicament, aquesta ultima fase de publicació consistiria en publicar el Plug-in de manera que qualsevol persona el pugui descarregar lliurement, i beneficiar-se dels seus avantatges.

Finalment, per concloure les conclusions, realitzar un incís respecte a la solució utilitzada en les reparacions no és la solució que m'agrada, però és la que s'ha decidit realitzar, perquè tota la lògica ha de residir en el servidor. Per poder realitzar el següent correctament s'haurien de generar els formularis en el client a partir d'un JSON. Aquesta tasca podria costar molt temps i, finalment, s'ha decidit utilitzar aquesta alternativa per poder finalitzar el projecte a temps i demostrar que és possible.

6. Possibles extensions

Les possibles extensions que sorgirien del present treball final de grau serien:

1.- Un dels possibles treballs que podria sorgir seria un treball que consistiria en la reestructuració i re-disseny, si és necessari, del codi del servidor. Si es vol aconseguir tenir una aplicació potent és important que els fonaments de l'aplicació estiguin ben preparats per poder sustentar la seva potencia i eficàcia. Els principals objectius d'aquest treball seria millorar la implantació del MVC en el servidor, preparar-lo per poder afegir funcionalitats de manera senzilla, és a dir, que es puguin implementar nous formularis per noves barreres. Crear una API RESTful seguint tots els estàndards que doni accés a totes les característiques necessàries per al desenvolupament de Plug-ins per a diverses plataformes.

2.- Un altre treball consistiria en continuar la feina de la doctora Afra Pascual, dissenyant formularis que permetessin reparar el màxim nombre de barreres que sigui possible en l'aplicació, oferint així una eina encara més útil.

3.- Un cop estigui el servidor consti amb una API perfecta, es pot millorar el client AngularJS realitzat en aquest treball per adaptar les API al nou format, realitzar les reparacions d'una manera correcta i millorar la integració d'aquest en l'entorn WordPress. A més es pot adaptar per a ser utilitzat en altres entorns CMS

7. Bibliografia

Access Control: CORS. (2016). Retrieved from Mozilla Developers Network: https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS

AngularJS. (2016). Retrieved from Google Corporation: https://angularjs.org/

AngularJS Documentation. (2016). Retrieved from Google Corporation: https://docs.angularjs.org/api

CORS. (2016). Retrieved from W3C: https://www.w3.org/TR/cors/

Documentation: Symfony. (2016, 06). Retrieved from SensioLabs: https://symfony.com/doc/current/index.html

MySQL. (2016). Retrieved from Oracle Corporation: https://www.mysql.com/

PHP Manual. (2016, 06). Retrieved from The PHP Group: http://php.net/manual/en/

WordPress. (2016). Retrieved from WordPress: https://wordpress.org/

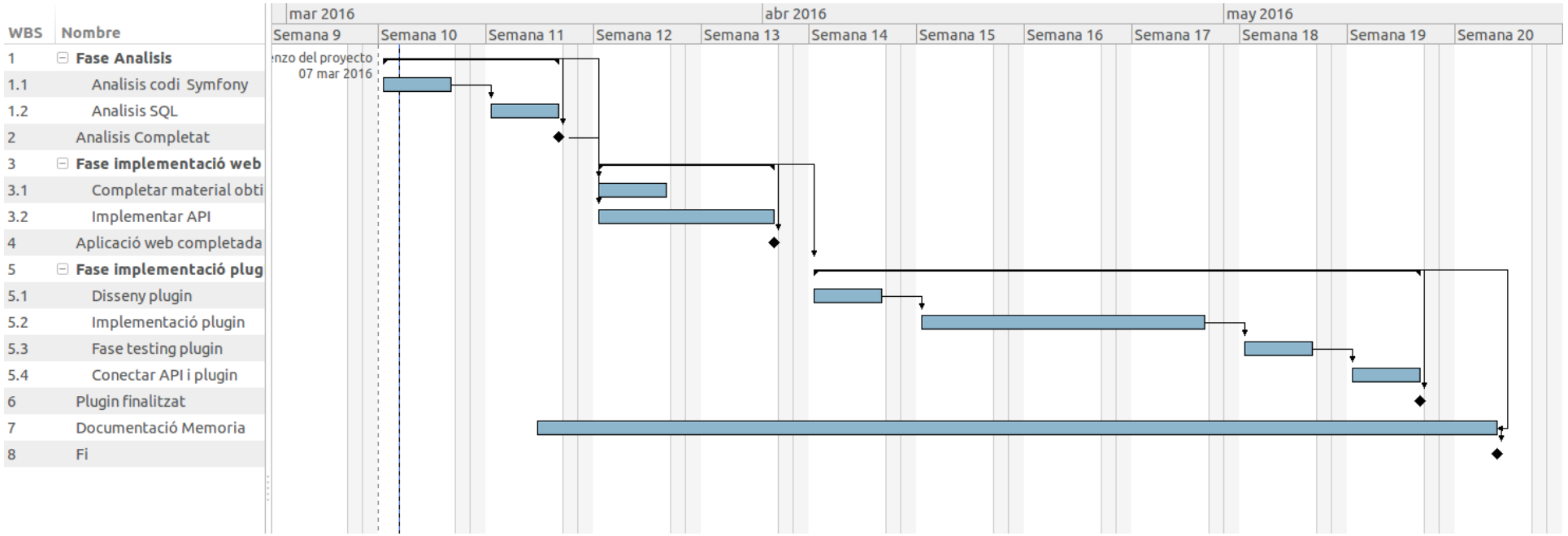
WordPress Developers. (2016). Retrieved from WordPress: https://codex.wordpress.org/Developer_Documentation

Eguiluz, J. (2012). *Desarrollo web ágil con Symfony2.*

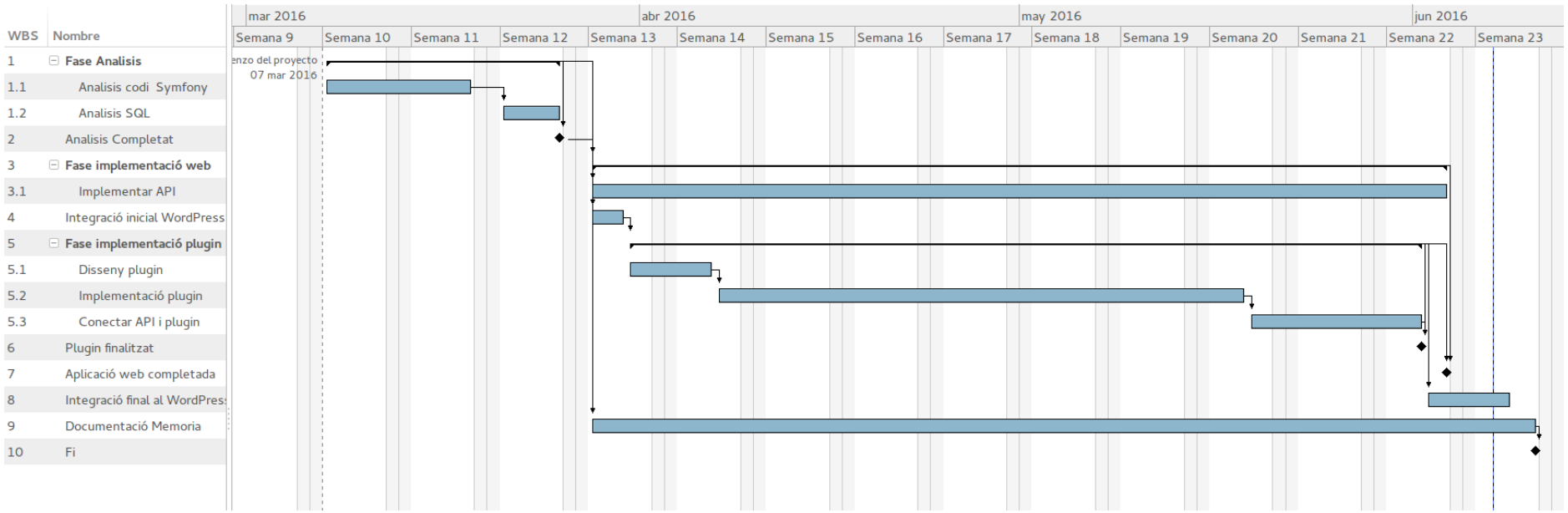
Pascual Almenara, A. (2015). Accesibilidad en entornos web interactivos: superación de las barreras digitales.

Annex 1

Planificació Inicial



Planificació Final



Base de dades